DIMACS Technical Report 97-16
April 1997

Optimization algorithms for separable functions with tree-like adjacency of variables and their application to the analysis of massive data sets

by

Ilya Muchnik
DIMACS, Rutgers University
P.O. Box 5062, New Brunswick, NJ 08903-5062, USA
muchnik@dimacs.rutgers.edu

Vadim Mottl
Tula State University, Lenin Ave. 92, 300600 Tula, Russia
mottl@atm.tsu.tula.ru

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs, Bellcore, and Bell Labs.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

A massive data set is considered as a set of experimentally acquired values of a number of variables each of which is associated with the respective node of an undirected adjacency graph that presets the fixed structure of the data set. The class of data analysis problems under consideration is outlined by the assumption that the ultimate aim of processing can be represented as a transformation of the original data array into a secondary array of the same structure but with node variables of, generally speaking, different nature, i.e. different ranges. Such a generalized problem is set as the formal problem of optimization (minimization or maximization) of a real-valued objective function of all the node variables. The objective function is assumed to consist of additive constituents of one or two arguments, respectively, node and edge functions. The former of them carry the data-dependent information on the sought-for values of the secondary variables, whereas the latter ones are meant to express the a priori model constraints. For the case when the graph of the pair-wise adjacency of the data set elements has the form of a tree, an effective global optimization procedure is proposed which is based on a recurrent decomposition of the initial optimization problem over all the node variables into a succession of partial problems each of which consists in optimization of a function of only one variable like Bellman functions in the classical dynamic programming. Two kinds of numerical realization of the basic optimization procedure are considered on the basis of parametric representation of Bellman functions, respectively, for discretely defined and quadratic node and edge functions. The proposed theoretical approach to the analysis of massive data sets is illustrated with its applications to the problems of the analysis of long molecular sequences and sets of geophysical explorative data.

1 Introduction

Despite the dramatic variety of application problems concerned with the necessity to analyze massive data arrays, it is possible to set off some single subclasses of problems which allow for treating them in unified terms of respective standard mathematical optimization problems for which there exist or can be created effective methods of solving.

One of such classes of data analysis problems is addressed in this paper. The class under consideration embraces, at least, massive sets of molecular sequences, satellite photographs as well as visual images of more general kind, sets of geophysical explorative data.

We will consider a data set $\mathbf{Y}$ as an experimentally acquired function y_t, $t \in T$, which takes values from an appropriate range $y \in \mathcal{Y}$, where index (argument) $t \in T$ runs through all the elements of the data set.

What is in common for the above mentioned kinds of data sets is the fact that the elementary information units y_t are ordered along the axes of one t or several arguments $\mathbf{t} = (t_1, \ldots, t_k)$, so that the data collection subject to an analysis as a whole is rather an array than set. Each element of such an array is naturally associated with a number of its neighbors which may be defined in several ways depending on the nature of the particular data source, but in any case a neighborhood relation on the set of data elements $t \in T$ is an inherent feature of the data array. It is convenient to express the neighborhood relation in the form of an undirected graph G which is understood as the set of neighboring element pairs $G \subset T \times T$.

In this work, we restrict our concern to only those of data array processing problems which can be represented as problems of transforming the original array $\mathbf{Y} = (y_t, t \in T)$, $y \in \mathcal{Y}$, into another function $\mathbf{X} = (x_t, t \in T)$, which would be defined on the same set of the array elements $t \in T$ and take values from a set $x \in \mathcal{X}$ specific for each particular problem. It is meant that each element x_t of the secondary data array offers an aggregated model of the hidden relationship of the identically indexed original element y_t to the entire array $\mathbf{Y}$.

In the problems of protein sequence analysis the original data array is a sequence of proteins $\mathbf{Y} = (\mathbf{y}_j,\ j = 1, \ldots, N)$ each of which, in its turn, is a sequence of amino acids $\mathbf{y}_j = (y_{ij},\ i = 1, \ldots, n_j)$. So, the data set to be analyzed has the structure of a two-dimensional array $\mathbf{Y} = (y_{\mathbf{t}}, \mathbf{t} = (i,j)) = (y_{ij},\ i = 1, \ldots, n_j,\ j = 1, \ldots, N)$ (Fig. 1).

As to the elements $x_{\mathbf{t}} = x_{ij}$ of the resulting array $\mathbf{X} = (\mathbf{x}_j,\ j = 1, \ldots, N) = (x_{ij},\ i = 1, \ldots, n,\ j = 1, \ldots, N)$, they might be sought for as vector variables $x_{ij} = (c_{ij}, z_{ij})$ (Fig. 2) taking values from an appropriate set $\mathcal{X}$. The first component c_{ij} is meant to have the sense of some averaged numerical characteristic of amino acids located at approximately the same position i in the proteins $\mathbf{y}_{j+\delta}$ which surround the respective protein $\mathbf{y}_j$ in the sequence to be analyzed. For instance, hydrophobicity could be taken as such a characteristic of each amino acid. The second component z_{ij} should point at the position in protein $\mathbf{y}_j$, including imaginary gaps between amino acids, where an

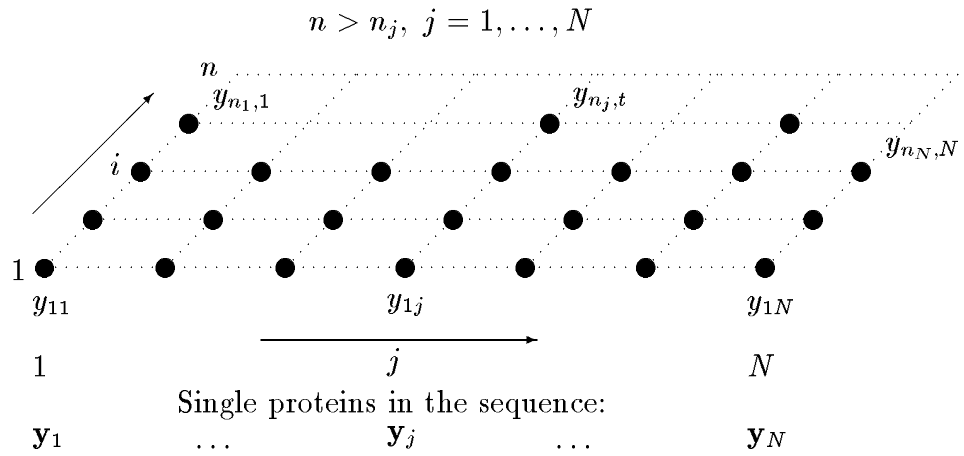

Figure 1: The original sequence of proteins as a two-dimensional data array.

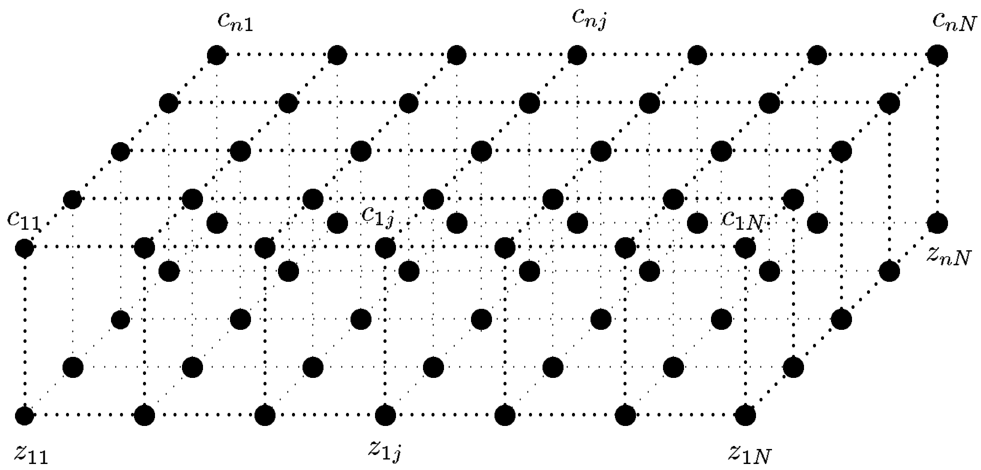

Figure 2: Two two-dimensional data arrays forming the result of multi-alignment of an ordered sequence of proteins.

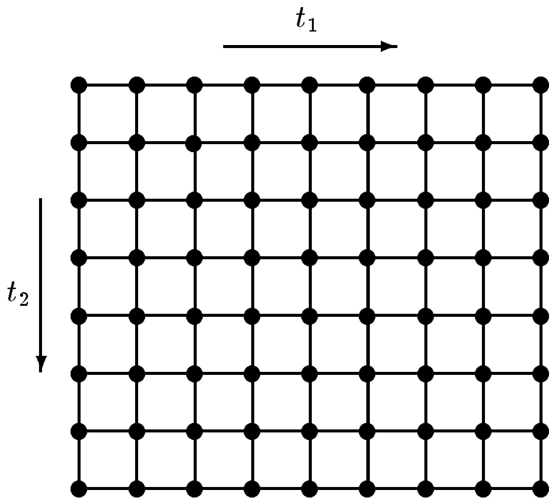

Figure 3: The two-dimensional image pixel grid and the simplest graph of pixel neighborhood in the form of a rectangular lattice.

amino acid with the respective property is expected to be located if judging from the surrounding of $\mathbf{y}_j$ in the entire protein sequence. This latter aspect of the result turns such an analysis into a version of the so-called multi-alignment of a set of proteins [1]. The version under consideration differs from the commonly adopted statement of the multi-alignment problem by the additional assumption that the set of the proteins is ordered, i.e. is a sequence. Below, in Section 4, we will discuss such a treatment of the multi-alignment problem in somewhat more detail.

In the problems of the analysis of visual images, in particular, *satellite photographs*, the original data set is a numerical function $\mathbf{Y} = (y_\mathbf{t},\, \mathbf{t} = (t_1, t_2) \in T)$, usually, brightness, which takes values from an appropriate set $y_\mathbf{t} \in \mathcal{Y}$ and is defined on the discrete two-dimensional pixel grid (Fig. 3). It is typical for low-level image processing problems that the desired result of processing is meant as another function $\mathbf{X} = (x_\mathbf{t},\, \mathbf{t} \in T)$ which would be defined on the same pixel grid $\mathbf{t} \in T$ and take values $x_\mathbf{t} \in \mathcal{X}$ from a set specific for each particular problem. We briefly consider here the specificity of three of such problems [8].

In the problem of segmentation of an original image into homogeneous regions of a preset number of classes, the class allocation of the image pixels will be naturally expressed by a variable $x_\mathbf{t}$ taking values from a finite set of class indices $\mathcal{X} = \{1, \dots, m\}$. So, the segmentation of the image into a set of classified regions can be sought for in the form of a class allocation map $\mathbf{X} = (x_\mathbf{t},\, \mathbf{t} \in T)$, $x_\mathbf{t} \in \mathcal{X} = \{1, \dots, m\}$, actually, a secondary finitely valued image.

In the problem of smoothing, the resulting function $\mathbf{X} = (x_\mathbf{t},\, \mathbf{t} \in T)$ is of the same nature as the original image, i.e. it is a picture. If we assume $\mathbf{X}$ to be real-valued $x_\mathbf{t} \in \mathsf{R}$, the set $\mathcal{X}$ the secondary picture takes values from remains the same $\mathcal{X} = \mathcal{Y} = \mathsf{R}$.

The problem of matching consists in finding identical points in two images of similar structure. Let two images $\mathbf{X}^l = (x^l_{\mathbf{t}},\ \mathbf{t} \in T)$ and $\mathbf{X}^r = (x^r_{\mathbf{t}},\ \mathbf{t} \in T)$ have the same range of definition T and form an entire image pair $\mathbf{X} = (\mathbf{X}^l, \mathbf{X}^r)$ to be analyzed as a whole, for instance a stereo pair of images, or photos of two human faces taken at the same scale, and the like. Whatever goal might be pursued by the analysis of a pair of similar images, the first and algorithmically most difficult stage is matching, i.e. establishing a correspondence in the two images between the point pairs which could be treated as identical. The result of such an operation is usually represented in the form of the so-called disparity map which, for each point $\mathbf{t}$ of the one of the images (conventionally taken as reference image), determines the vector $\mathbf{x}_{\mathbf{t}} = (x^1_{\mathbf{t}},\ x^2_{\mathbf{t}})$ indicating the spatial shift between $\mathbf{t}$ and its match $\mathbf{t}' = \mathbf{t} + \mathbf{x}_{\mathbf{t}}$ in the second (search) image. It is convenient to allow $\mathbf{x}_{\mathbf{t}}$ to take values from the two-dimensional real space $\mathcal{X} = \mathsf{R}^2$, and, thereby, permit inter-pixel interpolation in the discrete image grid T [9].

The geophysical explorative data sets, in other terms, so-called *seismic sections*, which are analyzed in the course of gas and oil reserves prospecting, consist of synchronous records of reflected seismic signals $\mathbf{y}_i = (y_{it},\ t = 1, \ldots, N)$ registered by a chain of geophones (seismic sensors) $i = 1, \ldots, n$ and initiated by a tentative seismic pulse. Being placed side by side, the seismograms form the vertical rows of a two-dimensional data array $\mathbf{Y} = (y_{\mathbf{t}},\ \mathbf{t} = (i, t) \in T$. Despite the fact that the initial argument t of seismic signals is time, a relevant preprocessing allows to identify the vertical axis with depth under the respective sensor, so that such a picture offers a vertical section through the under-surface medium. Local differences in the mode of spatial signal oscillations allow to judge about the hidden layered structure of the rock mass. As a rule, the aim of the analysis of a seismic section is its segmentation into a finite totality of homogeneous horizontally stretched regions interpreted as a geological model of rock layers. So, the result of data analysis is of the same kind as in the above mentioned problem of segmentation of visual images, i.e. it is a class allocation map $\mathbf{X} = (x_{\mathbf{t}},\ \mathbf{t} \in T)$, $x_{\mathbf{t}} \in \mathcal{X} = \{1, \ldots, m\}$.

In this work, we consider a generalized approach to such a class of massive data set analysis problems. The approach leans upon the natural and commonly adopted way to create algorithms of data analysis of practically any kind as those of optimization of a suitable objective function defined on a set which is meant to represent the whole variety of feasible results. For the class of problems under consideration, the result of data analysis is the sought-for secondary data array $\mathbf{X} = (x_{\mathbf{t}},\ \mathbf{t} \in T)$, so, it must occur as argument in an objective function $J(\mathbf{X}|\mathbf{Y})$ which should assess the discrepancy between each admissible version of the result $\mathbf{X}$ and the given data array $\mathbf{Y}$. The range of definition of the objective function is completely determined once the range of the elementary variables $x_{\mathbf{t}} \in \mathcal{X}$ is chosen. Thus, the result of data analysis $\hat{\mathbf{X}}(\mathbf{Y})$ is defined by the condition

$$\hat{\mathbf{X}}(\mathbf{Y}) = \arg \min_{x_{\mathbf{t}} \in \mathcal{X}} J(\mathbf{X}|\mathbf{Y}).$$

The principal concept of the approach consists in forming the objective function as a means of coordinating two kinds of information on the sought-for array $\mathbf{X}$, namely, the data-dependent local information on the value of each hidden variable $x_{\mathbf{t}}$, on the one hand, and the model-based constraints on the values of variables $(x_{\mathbf{t}'}, x_{\mathbf{t}''})$ adjacent to each other according to the accepted neighborhood graph $(\mathbf{t}', \mathbf{t}'') \in G$, on the other.

As a rule, the essence of the particular data analysis problem suggests a way to preliminary judge in a fuzzy form about the single value of the hidden variable $x_{\mathbf{t}}$ from a vicinity $\mathbf{Y}_{\mathbf{t}}$ of the respective point in the original array $\mathbf{Y}$. It is convenient to express such a judgment in the form of a data-dependent function $\psi_{\mathbf{t}}(x) = \psi(x|\mathbf{Y}_{\mathbf{t}})$ which must be defined over the set $\mathcal{X}$ and take the greater value the more evident is the contradiction between the hypothesis, that $x \in \mathcal{X}$ is just the "correct" local value we are seeking, and the current fragment $\mathbf{Y}_{\mathbf{t}}$ of the data array. Let such functions be called node functions, because each of them is associated with a node of the neighborhood graph on the set of data array elements.

However, practically inevitable interferences of various kinds usually prevent from taking the totality of locally optimal values

$$\hat{x}_{\mathbf{t}}(\mathbf{Y}_{\mathbf{t}}) = \arg \min_{x_{\mathbf{t}} \in \mathcal{X}} \psi(x_{\mathbf{t}}|\mathbf{Y}_{\mathbf{t}}), \quad \mathbf{t} \in T,$$

as the final decision on the sought-for secondary array $\hat{\mathbf{X}}(\mathbf{Y}) = (\hat{x}_{\mathbf{t}}(\mathbf{Y}_{\mathbf{t}}),\, \mathbf{t} \in T)$.

But the same essence of the particular application problem suggests, as a rule, also a model knowledge on the expected smoothness degree of the hidden data set. This knowledge can be often expressed in local terms as a model-based function $\gamma(x_{\mathbf{t}'},\, x_{\mathbf{t}''})$ which is defined over $\mathcal{X} \times \mathcal{X}$ at each pair of neighboring data array elements and takes the greater value the greater is the discrepancy, in accordance with the accepted smoothness model, between the respective neighboring hidden values. These functions will be referred to as edge functions, since each of them is associated with an edge of the neighborhood graph G.

Given a data array $\mathbf{Y}$, the decision on the secondary array $\mathbf{X} = (x_{\mathbf{t}},\, \mathbf{t} \in T)$ as a whole should be sought for as a compromise between the mutually contradictory requirements to minimize both node and edge functions. It is natural to take the sum

$$J(\mathbf{X}|\mathbf{Y}) = \sum_{\mathbf{t} \in T} \psi(x_{\mathbf{t}}|\mathbf{Y}_{\mathbf{t}}) + \sum_{(\mathbf{t}', \mathbf{t}'') \in G} \gamma(x_{\mathbf{t}'}, x_{\mathbf{t}''}) \tag{1}$$

as a combined criterion of decision making. As we see, the criterion to be minimized is a sum of items each of which is function of only one or two variables. The arguments of the criterion along with data-dependent functions $\psi_{\mathbf{t}}(x_{\mathbf{t}}) = \psi(x_{\mathbf{t}}|\mathbf{Y}_{\mathbf{t}})$ are associated with elements of the data array, i.e. with nodes of the neighborhood graph G (node functions), whereas the functions of two arguments $\gamma(x_{\mathbf{t}'}, x_{\mathbf{t}''})$ are assigned to edges of the graph (edge functions). We will call an objective function of such a kind to be separable, pairs of its variables occurring in summands $\gamma(x_{\mathbf{t}'}, x_{\mathbf{t}''})$ to be adjacent, and the graph G, that determines the structure of the function, its adjacency or supporting graph. We will also say that an objective function of kind (1) is a separable G-supported function.

The number of its variables is equal to the number $|T|$ of nodes in G, i.e. the number of elements in the data array.

In the problems of image analysis, the simplest graph of natural pixel neighborhood is rectangular lattice (Fig. 3). As to the problem of multi-alignment of ordered protein sequences, the suitable neighborhood graph, that will be considered in Section 4, is somewhat less straightforward, because the elements of the secondary data array (Fig. 2) are vector variables whose components interact in a specific manner.

The computational complexity of the minimization problem for a separable function substantially depends on the kind of the adjacency graph G. The fact itself that the objective function is separable does not contribute essentially to an improvement of the computational solvability of the optimization problem in comparison with that for an arbitrary function of many variables. An intent to create a non-iterative procedure of finding the global minimum point of a separable objective functions supported by a graph of general kind comes up against a fundamental barrier. In particular, the adjacency graph in Fig. 3 does not possess any specific properties from the viewpoint of the optimization problem.

But in the particular case, when graph G on the set T of variables has no cycles, i.e. is a tree, there exists a highly effective global optimization procedure based on a recurrent decomposition of the initial problem of the optimization of a function of $|T|$ variables into a succession of $|T|$ partial problems, each of which consists in the optimization of a function of only one variable.

In the next Section, we consider the principal structure of such a procedure in which the separable objective function is assumed to be supported by an arbitrary tree. In the particular case, when tree G has no branching nodes, i.e. is a chain, the procedure is nothing else than one of versions of the classical dynamic programming procedure. The partial functions of one variable to be minimized at each step of minimization of a tree-supported separable function play the same role as the Bellman functions in the classical procedure.

The procedure considered in this work, just like the classical dynamic programming procedure, runs only twice through all the nodes of the supporting tree. So, if we take the computational complexity of the global optimization of a function of one variable for unity, the optimization complexity for the objective function as a whole will be $|T|$-linear.

For this reason, it appears to be always preferable to set the problems of data analysis so that the separable objective functions to be minimized would be supported by trees. For the problems of image analysis, it was proposed in [8] to use several tree-supported objective functions to evaluate the goal variables at each of the vertical rows of the pixel grid of an image, conventionally, the vertical ones $\mathbf{x}_{t_1^*} = (x_{t_1^*,t_2},\ t_2 = 1,\dots,N_2)$ (Fig. 4):

$$\hat{\mathbf{x}}_{t_1^*} = \arg \min_{x_{t_1^*,t_2} \in \mathcal{X}} \left\{ \min_{x_{t_1,t_2} \in \mathcal{X},\, t_1 \neq t_1^*} J_{t_1^*}(\mathbf{X}) \right\},\ \ t_1^* = 1,\dots,N_1.$$

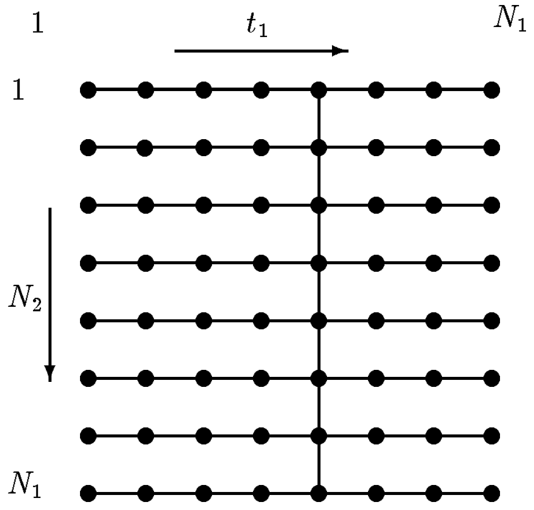

Figure 4: Neighborhood tree for evaluating the goal variables at a single vertical row of the pixel grid.

In Section 4, we will show the way how such an approach can by applied to the problem of multi-alignment of an ordered protein sequence.

2 Optimization of separable functions supported by trees

2.1 The structure of the objective function. Node and edge functions

In this Section, we don't associate the variables of the objective function, i.e. the nodes of the supporting tree, with variables in any certain application problem. To denote them, we will use symbol t instead of two symbols i and j in the problem of multi-alignment of proteins and denotation $\mathbf{t} = (t_1, t_2)$ in the generalized image processing problem.

Let G be a tree on the set of nodes $T = \{t\}$ (Fig. 5), and

$$J(X) = \sum_{t \in \mathrm{T}} \psi_t(x_t) + \sum_{(t', t'') \in G} \gamma_{t',t''}(x_{t'}, x_{t''}), \quad x_t \in \mathcal{X}_t, \tag{2}$$

be a G-supported separable objective function to be minimized in an arbitrary collection of ranges of variables. Functions of one variable $\psi_t(x_t)$ at nodes of the graph will be called node functions, and functions of two variables $\gamma_{t',t''}(x_{t'}, x_{t''})$ assigned to edges will be, respectively, called edge functions.

It is out of significance which of the nodes t^* is assigned the root of the tree, but once it is done, the set of nodes gets partitioned into a number of subsets each consisting of nodes of a certain level

$$T = \bigcup_{j=1}^{M} T^j, \quad T^j \bigcap T^k = \emptyset, \quad j \neq k.$$

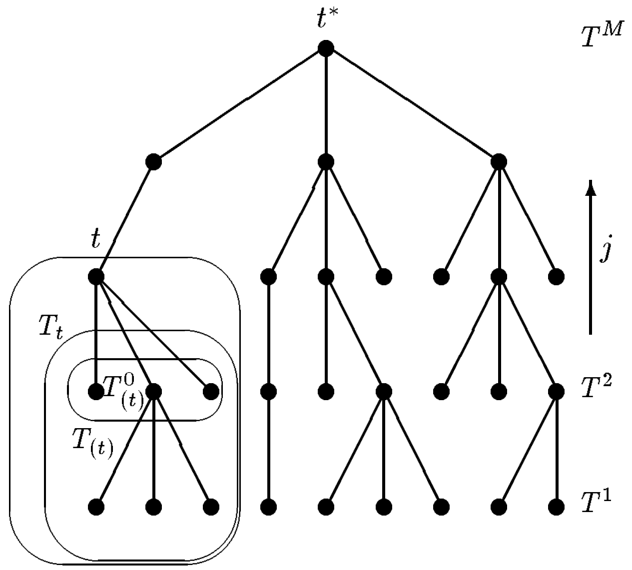

Figure 5: Structure of an arbitrary tree.

We will denote the set of nodes of the lowest level by T^1, and the set containing the only node of the highest level, namely, the root, by T^M.

Let, further, the tree $G_t \subset G$ formed by a node t and its descendants be called descendant tree of this node. It is clear that $G_t = \emptyset$ for leaves, in particular, for all $t \in T^1$, and $G_t = G$ for the root $t \in T^M$. The set of all nodes in the descendant tree of a node t will be denoted by T_t, symbol $T_{(t)}$ will mean the same set without this node itself, i.e. the set of its descendants, and $T^0_{(t)} \subseteq T_{(t)}$ will denote the subset of its immediate (next) descendants. So, $T_t = \{t\}$ and $T^0_{(t)} = T_{(t)} = \emptyset$ for leaves, and $T_{t*} = T$ for the root.

Analogously, symbols $X_t = (x_s,\ s \in T_t)$ and $X_{(t)} = (x_s,\ s \in T_{(t)})$ will mean partial vectors of variables at the respective sets of nodes.

Finally, let us associate with each node t a partial objective function

$$J_t(X_t) = J_t(x_t, X_{(t)}) = \sum_{s \in T_t} \psi_s(x_s) + \sum_{(s',s'') \in G_t} \gamma_{s',s''}(x_{s'}, x_{s''})$$

which has absolutely the same structure as the full objective function (2) but is defined on the respective partial descendant tree. It is easy to see that

$$J_t(X_t) = J_t(x_t, X_{(t)}) = \psi_t(x_t) + \sum_{s \in T_{(t)}} \gamma_{t,s}(x_t, x_s) + \sum_{s \in T_{(t)}} J_s(X_s) =$$

$$\psi_t(x_t) + \sum_{s \in T_{(t)}} \Big\{ \gamma_{t,s}(x_t, x_s) + J_s(x_s, X_{(s)}) \Big\}.$$

In particular, for the nodes of the lowest level $t \in T^1$, as well as for all other leaves, the partial objective function gets trivial $J_t(X_t) = J_t(x_t) = \psi_t(x_t)$, and for the root the partial function is nothing else than the full objective function $J_{t^*}(X_{t^*}) = J(X)$ (2).

2.2 Bellman functions and the basic optimization procedure

The principal idea of the optimization procedure for separable objective functions supported by trees is based, like the classical dynamic programming procedure for chains, on the notion of Bellman function

$$\tilde{J}_t(x_t) = \min_{X_{(t)}} J(x_t, X_{(t)}) = \psi_t(x_t) + \min_{X_s,\, s \in T^0_{(t)}} \sum_{s \in T^0_{(t)}} \{\gamma_{t,s}(x_t, x_s) + J_s(X_s)\}, \quad x_t \in \mathcal{X}.$$

The fundamental property of the Bellman function

$$\tilde{J}_t(x_t) = \psi_t(x_t) + \sum_{s \in T^0_{(t)}} \min_{x_s \in \mathcal{X}} \left\{\gamma_{t,s}(x_t, x_s) + \tilde{J}_s(x_s)\right\} \tag{3}$$

will be called the upward recurrent relation. The inverted form of this relation

$$\tilde{x}_s(x_t) = \arg\min_{x_s \in \mathcal{X}} \left\{\gamma_{t,s}(x_t, x_s) + \tilde{J}_s(x_s)\right\}, \quad s \in T^0_{(t)}, \tag{4}$$

will be referred to as the downward recurrent relation.

The optimization procedure is based on the assumption that there exists a sufficiently effective way of solving the partial optimization problems occurring in (3), and storing, in a compact form, the Bellman functions or downward relations.

The procedure runs twice through all the nodes of the graph in order of their membership in the hierarchy levels T^j, first upwards $j = 1, \dots, M$, and then downwards $j = M, \dots, 2$.

It starts with the nodes of the lowest (first) hierarchy level $t \in T^1$, $j = 1$. As $T^0_{(t)} = \emptyset$ for these nodes,

$$\tilde{J}_t(x_t) = \psi_t(x_t), \quad x_t \in \mathcal{X}, \quad t \in T^1. \tag{5}$$

Thereupon, the upward recurrent relation (3) is applied sequentially to the nodes of the higher levels $j = 2, \dots, M$. The Bellman functions $\tilde{J}_t(x_t)$ or downward relations $\tilde{x}_s(x_t)$, $x_t \in \mathcal{X}$, are to be stored for all the nodes of the tree except those of the first level.

The Bellman function at the root $\tilde{J}_{t^*}(x_{t^*})$, $x_{t^*} \in \mathcal{X}$, that is obtained at the last step of the upward pass, immediately gives the optimal value of the root variable:

$$\hat{x}_{t^*} = \arg\min_{x_{t^*} \in \mathcal{X}} \tilde{J}_{t^*}(x_{t^*}). \tag{6}$$

On the downward pass, as the procedure descends the hierarchy levels $j = M, \dots, 2$, the already found optimal values of the current level determine optimal values of the variables at the immediately underlying level in accordance with the downward recurrent rules (4) stored as the result of the upward pass or determined by the stored Bellman functions (3)

$$\hat{x}_s = \tilde{x}_s(\hat{x}_t), \quad s \in {T_{(t)}}^0. \tag{7}$$

The degenerate particular case of a tree is tree without branching nodes, i.e. a chain. In this case, the above described optimization procedure becomes the classical dynamic programming procedure.

We introduce here an additional denotation $\hat{J}_t(x_t)$ for the so-called marginal node functions each of which shows how the full objective function depends on the value of a single variable at the respective node t, if other node variables take conditionally optimal values in accordance with x_t:

$$\hat{J}_t(x_t) = \min_{x_s,\, s\in T,\, s\neq t} J(X). \tag{8}$$

At the root, the marginal node function is found at the last step of the upward pass $\hat{J}_{t^*}(x_{t^*})$. It can be shown that the marginal node functions for other nodes satisfy the downward recurrent relation

$$\hat{J}_s(x_s) = \min_{x_t\in\mathcal{X}} \left\{ \hat{J}_t(x_t) + \left[\tilde{J}_s(x_s) + \gamma_{t,s}(x_t, x_s)\right] - \min_{x'_s\in\mathcal{X}} \left[\tilde{J}_s(x'_s) + \gamma_{t,s}(x_t, x'_s)\right] \right\}, \quad s \in T^0_{(t)}. \tag{9}$$

3 Numerical realization of the optimization procedure for discretely defined and quadratic node and edge functions

A numerical realization of the basic procedure for a separable function supported by a tree, just like the classical dynamic programming procedure, is possible only in case there exists a finitely parameterized function family $\tilde{J}(x;\mathbf{a})$ concordant with node functions $\psi_t(x)$ and edge functions $\gamma_{t',t''}(x', x'')$ in the sense that Bellman functions (3) belong to this family at each step. In this case, the upward pass of the procedure consists in a recurrent re-evaluating parameters $\tilde{\mathbf{a}}_t$ that completely represent the Bellman functions.

We consider here two kinds of such a parameterization which present, in some sense, two extremes of the range of possible ways to define node and edge functions.

The evident case of a trivial parameterization is the case of a discrete range of node variables $\mathcal{X} = \{1, \dots, m\}$. In this case, Bellman functions (3) are m-dimensional vectors, downward relations (4) are matrices $(m \times m)$, and the elementary enumeration gives an easy solution of the partial optimization problems.

Another extreme case is the assumption that the range of node variables is n-dimensional real space $\mathbf{x} \in \mathcal{X} = \mathsf{R}^n$, and both node and edge functions are quadratic with positive definite Hessian matrices:

$$\begin{aligned} \psi_t(\mathbf{x}) &= (\mathbf{x} - \mathbf{x}^0_t)^T \mathbf{Q}^0_t (\mathbf{x} - \mathbf{x}^0_t), \\ \gamma_{t',t''}(\mathbf{x}', \mathbf{x}'') &= (\mathbf{x}' - \mathbf{x}'')^T \mathbf{U}_{t',t''} (\mathbf{x}' - \mathbf{x}''). \end{aligned} \tag{10}$$

Then, each of the Bellman functions will be likewise quadratic

$$\tilde{J}_t(\mathbf{x}_t) = \text{const} + (\mathbf{x} - \tilde{\mathbf{x}}_t)^T \tilde{\mathbf{Q}}_t (\mathbf{x} - \tilde{\mathbf{x}}_t), \quad \mathbf{x} \in \mathsf{R}^n, \tag{11}$$

and, so, completely represented by its minimum point $\tilde{\mathbf{x}}_t \in \mathsf{R}^n$ and positive definite Hessian matrix $\tilde{\mathbf{Q}}_t\,(n \times n)$, because constant items are out of significance for the optimization result. It follows from (3) and (11) that

$$\tilde{\mathbf{Q}}_t = \mathbf{Q}_t^0 + \sum_{s \in T^0_{(t)}} \left[\tilde{\mathbf{Q}}_s^{-1} + \mathbf{U}_{t,s}^{-1}\right]^{-1}, \quad \tilde{\mathbf{x}}_t = \tilde{\mathbf{Q}}_t^{-1} \left\{ \mathbf{Q}_t^0 \mathbf{x}_t^0 + \sum_{s \in T^0_{(t)}} \left[\tilde{\mathbf{Q}}_s^{-1} + \mathbf{U}_{t,s}^{-1}\right]^{-1} \tilde{\mathbf{x}}_s \right\}. \tag{12}$$

Analogously, it follows from (4), (10) and (12) that the downward recurrent relations will be linear

$$\tilde{\mathbf{x}}_s(\mathbf{x}_t) = (\mathbf{I} - \tilde{\mathbf{H}}_s)\tilde{\mathbf{x}}_s + \tilde{\mathbf{H}}_s \mathbf{x}_t = \tilde{\mathbf{x}}_s + \tilde{\mathbf{H}}_s(\mathbf{x}_t - \tilde{\mathbf{x}}_s), \quad s \in T^0_{(t)}, \tag{13}$$

and, so, completely representable by their transition matrices

$$\tilde{\mathbf{H}}_s = (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1}\mathbf{U}_{t,s} = \left[\mathbf{U}_{t,s}^{-1}\tilde{\mathbf{Q}}_s + \mathbf{I}\right]^{-1}. \tag{14}$$

The first (upward) pass of the procedure starts with the initial assignment in accordance with (5) and (10) $\tilde{\mathbf{x}}_t = \mathbf{x}_t^0$, $\tilde{\mathbf{Q}}_t = \mathbf{Q}_t^0$, $t \in T^1$, $j = 1$, whereupon vectors $\tilde{\mathbf{x}}_t$ and matrices $\tilde{\mathbf{Q}}_t$ are recurrently recalculated, according to (12), for $t \in T^j$ at levels $j = 2, \ldots, M$ and stored. The last step $j = M$ gives the final result for the root $\hat{\mathbf{x}}_{t^*} = \tilde{\mathbf{x}}_{t^*}$ (6). On the second (downward) pass $t \in T^j, j = M, \ldots, 2$, the set of nodes, for which the final values of variables are already found, propagates downwards in accordance with the recurrent rule

$$\hat{\mathbf{x}}_s = \tilde{\mathbf{x}}_s(\hat{\mathbf{x}}_t), \quad s \in T^0_{(t)}, \tag{15}$$

that follows from (13) and (14).

All the marginal functions will be also quadratic

$$\hat{J}_t(\mathbf{x}_t) = \text{const} + (\mathbf{x}_t - \hat{\mathbf{x}}_t)^T \hat{\mathbf{Q}}_t (\mathbf{x}_t - \hat{\mathbf{x}}_t),$$

where minimum points are given by the downward recurrent rule (15). It can be shown that the recurrent rule (9) leads to the following downward procedure of recalculating the marginal Hessian matrices:

$$\hat{\mathbf{Q}}_s = \left[\tilde{\mathbf{H}}_s \hat{\mathbf{Q}}_t^{-1} \tilde{\mathbf{H}}_s^T + \left(\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s}\right)^{-1}\right]^{-1}, \quad s \in T^0_{(t)}. \tag{16}$$

At the root, the marginal node function is found at the last step of the upward pass $\hat{\mathbf{Q}}_{t^*} = \tilde{\mathbf{Q}}_{t^*}$ (12).

4 Remarks on multi-sequence alignment of proteins

4.1 The problem of multi-alignment of an ordered protein sequence

In the problem of multi-sequence alignment of proteins, the original data set is a sequence $\mathbf{Y} = (\mathbf{y}_j,\ j = 1, \ldots, N)$ of proteins each of which, in its turn, is represented by the respective sequence of amino acid residues $\mathbf{y}_j = (y_{ij},\ i = 1, \ldots, n_j)$, where symbols y take values from the set of 20 amino acids usually denoted by capital Latin letters:

$$\mathcal{Y}^0 = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}.$$

So, the data set to be analyzed has the structure of a two-dimensional array (Fig. 1). The problem of multi-alignment is underlaid by the supposition that single proteins in the set

$$\mathbf{Y} = (\mathbf{y}_j,\ j = 1, \ldots, N) = (y_{ij},\ i = 1, \ldots, n_t,\ j = 1, \ldots, N) \tag{17}$$

are not fatally different, i.e. their amino acid sequences $\mathbf{y}_{j'} = (y_{ij'},\ i = 1, \ldots, n_{j'})$, $\mathbf{y}_{j''} = (y_{ij''},\ i = 1, \ldots, n_{j''})$ are quite similar to each other and can be mutually interconverted by a relatively small number of insertions, deletions, and replacements of amino acids in each of them, in other words, aligned, brought to a common length. Therefore, it is convenient to consider the extended set of amino acids $\mathcal{Y} = \mathcal{Y}^0 \cup \{\Delta\}$ that includs the additional null symbol Δ which will denote the eventual gap between two actual amino acids in an alignment $\mathcal{Y} = \mathcal{Y}^0 \cup \{\Delta\}$.

The aim of multi-alignment is to show to the researcher what is in common in the analyzed set of proteins from the viewpoint of their amino-acid makeup. There exist several approaches to the problem of multi-alignment, but we refer here only to the pioneer work [1]. In contrast to the known statements of the problem, we distinguish between ordered and unordered protein sets, and, according to this choice, treat indices $j = 1, \ldots, N$ as either ordinal numbers or mere labels of distinct proteins in the set subject to an analysis as a whole. We address here, primarily, the problem of multi-alignment of ordered protein sequences, which has not yet been discussed in the literature. As to the problem of multi-alignment of unordered sets of sequences, it will be considered as a particular case of the "ordered" version.

We will keep to the strategy of multi-alignment based on the notion of pattern sequence $\mathbf{c} = (c_i,\ i = 1, \ldots, n)$ which is meant to express the common properties of a set of proteins. The elements of the pattern sequence take values from an appropriate set $c \in \mathcal{C}$ that may be different from the set of original amino-acid residues and have the sense, for instance, of frequency distributions over $\mathcal{Y}$ [2, 5], subclasses in it [6], or a functional descriptions of amino acid properties [2, 7]. However, the principal assumption of this work, that proteins forming the data set $\mathbf{Y} = (\mathbf{y}_j,\ j = 1, \ldots, N)$ are ordered into a chain, requires a substantial revision of the very notion of multi-alignment. The necessity of a revision is concerned with the reason that, in such a chain, it appears to be senseless to align proteins $\mathbf{y}_{j'}$ and $\mathbf{y}_{j''}$ with quite distant indices j' and j''.

In this work, we will assign each protein $\mathbf{y}_j$ its own pattern sequence $\mathbf{c}_j = (c_{kj}, k = 1,\ldots,n)$ of a fixed length n along with its individual alignment $\mathbf{z}_j = (z_{kj}, k = 1,\ldots,n)$ to this sequence. The common length of the pattern sequences should be taken somewhat greater than that of the longest amino acid sequence in the protein set to be analyzed $n > \max\{n_j,\ j = 1,\ldots,N\}$. The individual alignment of a protein $\mathbf{y}_j$ will have the form of a sequence $\mathbf{z}_j = (z_{kj}, k = 1,\ldots,n)$ in which each element z_{kj} is associated with the respective pattern sequence element c_{kj} and points at the position of its match in the protein $\mathbf{y}_j = (y_{ij},\ i = 1,\ldots,n_j)$, including possible gaps between neighboring aminoacids $y_{i-1,j}$ and $y_{i,j}$, under some additional restrictions aimed at preventing crossed references.

So, the complete result of data analysis consists of two arrays (Fig. 2)

$$\mathbf{C} = (\mathbf{c}_j,\ j = 1,\ldots,N) = (c_{ij},\ i = 1,\ldots,n,\ j = 1,\ldots,N), \tag{18}$$

$$\mathbf{Z} = (\mathbf{z}_j,\ j = 1,\ldots,N) = (z_{ij},\ i = 1,\ldots,n,\ j = 1,\ldots,N), \tag{19}$$

each of which retains the two-dimensional structure (Fig. 1) of the original data set (17). Jointly, these two arrays make a three-dimensional array which offers a succession of generalized description of all the proteins, where each protein is described by two sequences, namely, the pattern and alignment one.

It may turn out to be is convenient to denote the pattern sequence coupled with the alignment of the respective protein to it by combined variable $\mathbf{x}_j = (\mathbf{c}_j,\ \mathbf{z}_j)$, $\mathbf{x}_j = (x_{ij},\ i = 1,\ldots,n) = (c_{ij}, z_{ij}\ i = 1,\ldots,n)$, where $x = (c,z) \in \mathcal{X} = \mathcal{C} \times \mathcal{Z}$. In these terms, the result of data analysis is succession

$$\mathbf{X} = (\mathbf{x}_j,\ j = 1,\ldots,N) = (x_{it},\ i = 1,\ldots,n,\ j = 1,\ldots,N) \tag{20}$$

that forms an array of the same structure as the succession of original proteins.

Actually, the problem of multi-alignment of an ordered set of proteins consists in finding such a succession of pattern sequences $\mathbf{C} = (\mathbf{c}_1,\ldots,\mathbf{c}_N)$, in which single sequences $\mathbf{c}_j$ would be as close as possible as to respective proteins $\mathbf{y}_j$ in the sense of the best individual alignments $\mathbf{z}_j$, as well as to the precedent sequences $\mathbf{c}_{j-1}$.

Thus, the resulting succession of pattern sequences and individual alignments is sought-for as a "smooth" approximation to the original succession of proteins in somewhat different terms, because elements of pattern sequences are no longer actual amino acids, rather, generalized descriptions of subclasses of them. Such a smoothing is aimed at highlighting eventual hidden trends or even abrupt changes in the protein succession.

4.2 Objective functions of multi-alignment

In accordance with the commonly adopted practice in molecule sequence analysis, it appears to be natural to formalize the problem of multi-alignment of an ordered sequence of proteins as an optimization problem with an appropriate objective function. We consider here two versions of the problem statement, a straightforward statement with a separable objective function of general kind, and a more sophisticated one, which will lead to several separable objective functions each supported by a tree. It will be just the second version that will allow to use all the computational advantages which, as it was shown in Section 2, follow from the fact that the separable function to be minimized is supported by a tree.

The first version of the problem statement presupposes minimization of only one objective function whose minimum point immediately gives the sought-for solution on the values of all the variables:

$$(\hat{\mathbf{C}}, \hat{\mathbf{Z}}) = \arg \min_{c_{ij} \in \mathcal{C}, z_{ij} \in \mathcal{Z}} J(\mathbf{C}, \mathbf{Z}),$$

$$J(\mathbf{C}, \mathbf{Z}) = \sum_{j=1}^{N} \sum_{i=1}^{n} \psi_{it}(c_{ij}) + \sum_{j=1}^{N} \sum_{i=2}^{n} \gamma(c_{i-1,j}, c_{ij}) + \sum_{j=2}^{N} \sum_{i=1}^{n} \mu(z_{i-1,j}, z_{ij}), \tag{21}$$

$$\psi_{ij}(c) = d(c, y_{ij}),$$

where $d(c, y)$, $c \in \mathcal{C}$, $y \in \mathcal{Y}$, is a preset loss function of replacing amino acid y by pattern element c, function $\gamma(c', c'')$ defines a distance measure between elements of two neighboring pattern sequences, and $\mu(z', z'')$ is penalty function restricting admissible neighborhood of shifts in an alignment.

In the second version, the generalized description of each protein $(\mathbf{c}_{j^*}, \mathbf{z}_{j^*}) = (c_{ij^*}, z_{ij^*},\ i = 1, \dots, n)$, $j^* = 1, \dots, N$, with respect to the entire data set $(\mathbf{C}, \mathbf{Z})$ is sought-for as the minimum point of an individual objective function

$$(\hat{\mathbf{c}}_{j^*}, \hat{\mathbf{z}}_{j^*}) = \arg \min_{c_{ij^*} \in \mathcal{C}, z_{ij^*} \in \mathcal{Z}} \left\{ \min_{c_{ij} \in \mathcal{C}, z_{ij} \in \mathcal{Z}, j \neq j^*} J_{j^*}(\mathbf{C}, \mathbf{Z}) \right\}, \quad j^* = 1, \dots, N,$$

$$J_{j^*}(\mathbf{C}, \mathbf{Z}) = \sum_{j=1}^{N} \sum_{i=1}^{n} \psi_{ij}(c_{ij}) + \sum_{i=2}^{n} \gamma(c_{i-1,j^*}, c_{ij^*}) + \sum_{j=2}^{N} \sum_{i=1}^{n} \mu(z_{i,j-1}, z_{ij}). \tag{22}$$

Fig. 6 shows the structure of each of the two objective functions in the form of an undirected graph whose nodes are associated with arguments of the function, and edges specify pairs of them which occur as arguments in particular summands. Let this graph be denoted by G and understood as set of such variable pairs $\mathsf{G} \subset \mathsf{T} \times \mathsf{T}$, where T is the set of all the variables of the objective function as c_{ij}, as well as z_{ij}, and, thus, contains twice as many elements as set $T = \{ij\}$, what is clearly seen when comparing Fig. 2 and Fig. 3.

As we see, the graphs for the partial objective functions in the second version of the problem statement (22) are trees (Fig. 6,b) in contrast to the first version (21) for which the supporting graph inevitably has cycles (Fig. 6,a). It is just the second version of the statement of the multi-alignment problem which allows for creating an effective optimization algorithm on the basis of the optimization procedure proposed in Section 2.

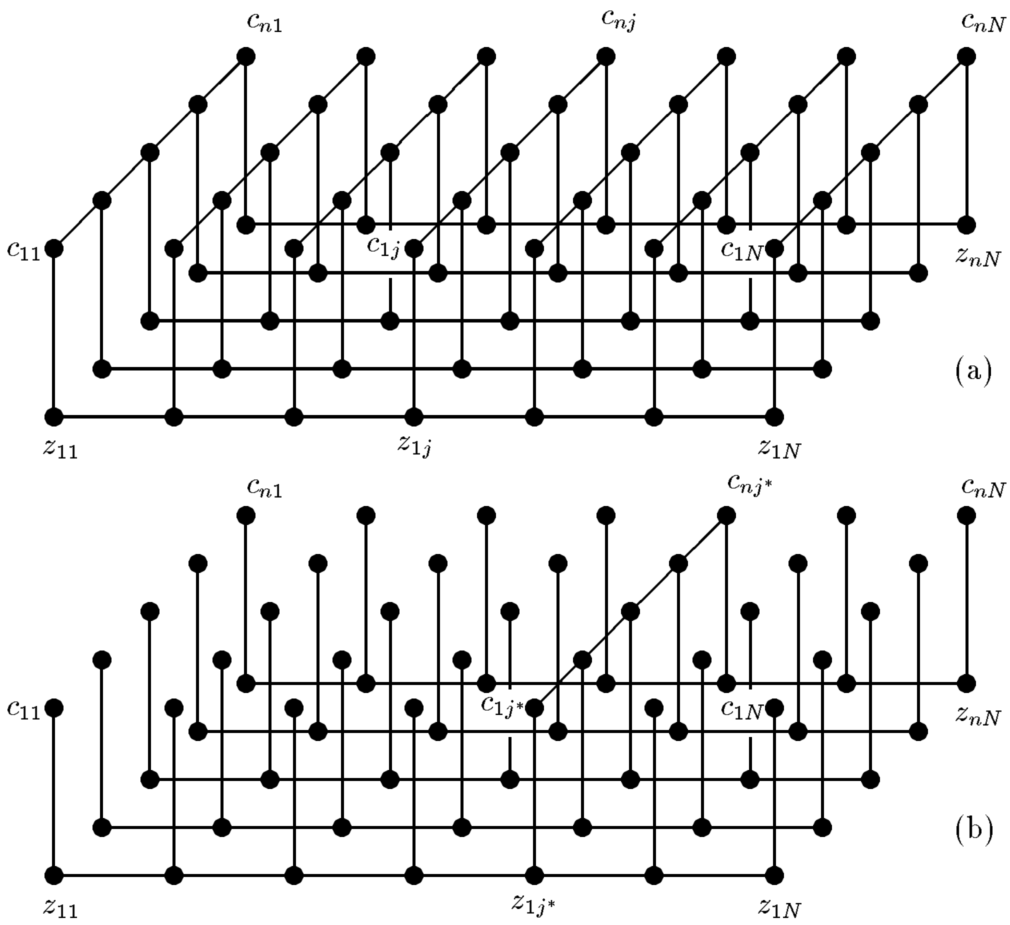

Figure 6: Graphs of the adjacency of variables for the first (a) and second (b) version of the separable objective function in the problem of protein sequence multi-alignment.

References

[1] Waterman, M.S., Smith, T.F., and Beyer, W.A. Some biological sequence metrics. *Advanced Mathematics*, 20, 1976, pp. 367-387.

[2] Taylor, W.R. Identification of protein sequence homology by consensus template alignment. *Journal of Molecular Biology*, 188, 1986, pp. 233-258.

[3] Taylor, W.R. A flexible method to align large numbers of biological sequences. *Journal of Molecular Evolution*, 28, 1988, pp. 161-169.

[4] Mirkin, B., and Roberts, F.S. Consensus functions and patterns in molecular sequences. *Technical Report* 92-37. DIMACS, Rutgers University, August, 1992.

[5] Gribskov, M., McLachlan A.D., Eisenberg, D. Profile analysis: detection of distantly related proteins. *Proc. Nat. Acad. Sci. USA*, 84, 1987, pp. 4355-4358.

[6] Smith, R.F., and Smith, T.F. Pattern-induced multi-sequence alignment (PIMA) algorithm employing secondary structure-dependent gap penalties for use in comparative protein modeling. *Protein Engineering*, 5, 1992, pp. 35-41.

[7] Bashford, D., Chothia, C., and Lesk, A.M. Determinants of a protein fold. *Journal of Molecular Biology*, 196, 1987, pp. 199-216.

[8] Mottl, V.V., Muchnik, I.B., Blinov, A.B., and Kopylov, A.V. Hidden tree-like quasi-Markov model and generalized technique for a class of image processing problems. *13th International Conference on Pattern Recognition.* Vienna, Austria, August 25-29, 1996. Track B, pp. 715-719.

[9] Mottl, V.V., Kopylov, A.V., Blinov, A.B., and Zheltov, S.Yu. Quasi-statistical approach to the problem of stereo image matching. *Proceedings SPIE*, 2363, 1994, pp. 50-61.